Appropriateness of Transport Mechanisms in Data Grid Middleware

Rajkumar Kettimuthu Sanjay Hegde William E. Allcock John Bresnahan

Mani Potnuru

Mathematics and Computer Science Division
Argonne National Laboratory
{*kettimut,hegdesan,allcock,bresnaha,potnuru*}*@mcs.anl.gov*

1 Problem Statement

Bulk data transfer has become one of the key requirements in many Grid applications. GridFTP has been widely deployed for high-speed data transport services. These services normally require reliable data transfer resulting in TCP as the preferred common base protocol. Unfortunately TCP performs sub-optimally in achieving maximum throughput on the currently available "long fat networks" over the Internet.

This issue required two phases of investigation - first, appropriate instrumentation of TCP stack and studying the instrumented standard Linux TCP stack. A TCP stack incorporating the recently proposed modifications for high-speed transport will accompany this. The second phase will consider non-TCP based reliable transport mechanisms. Initially we have selected an algorithm known as NETBLT, a reliable mechanism based on UDP. There are several other alternatives to NETBLT that will be investigated in future work. These include Tsunami from Indiana University, RBUDP from the Electronic Visualization Lab at University of Illinois Chicago.

2 Approach

We propose a mechanism called SNACK (Selective Negative Acknowledgment) to improve the performance of TCP in the face of multiple dropped segments. The basic idea behind this approach is to notify the sender about the segments that have been received successfully by sending information about the missing segments. By sending selective negative acknowledgments, the receiver of data informs the sender about a list of missing segments and the sequence number of the segment up to which there is no lost segment (other than the ones mentioned in the list of missing segments); hence the sender need not retransmit the segments that have reached the receiver already. Thus, SNACK can convey more information than SACK with same number of bits in some situations.

We also propose an adaptive scheme called DynACK that dynamically switches between SACK and SNACK based on the number of bits required by SACK or SNACK to represent segments that have been received successfully. The receiver chooses the one that requires fewer bits to convey more information. The receiver then sets a bit in the DynACK option to let the sender know whether it uses SACK or SNACK. By examining that bit, the sender appropriately uses SACK/SNACK to mark the segments that have reached the receiver but not yet acknowledged.

Recently a set of enhancements has been proposed to address the above-mentioned problems for high-speed TCP, such as Quick Start for TCP/IP, Limited Slow Start and High Speed Response Function. Currently, some of these mechanisms have been evaluated in a simulated environment. But in order to standardize these schemes, more experiments are needed to fully evaluate their performance in real environments. In this work, we study the performance of the above mentioned enhancements for bulk data transfer.

We employ an API, which supports pluggable transport drivers called globus_XIO. The functions defined in the interface are implemented, which allow the higher-level test scripts to run unchanged using the new protocol. These tests are then run using web 100 to provide detailed statistics of the TCP stack operations. Graphs indicating key parameters such as TCP

window size and the effect of events such as dropped packets on these parameters will provide insight into the operation during bulk transport. This is particularly worthy of note during multi-stream transport. These tests are repeated on an instrumented Linux kernel incorporating modifications for improved behavior during bulk data transport on long fat network pipes. We also present head to head performance comparisons of various configurations.

3 Visual Presentation

The poster will consist of a simple schematic representation of our testing environment; graphs showing TCP stack behavior, graphs of relative performance of the various algorithms, a discussion of the pros and cons, and finally a proposal for future work.